Parveen Khurana

124 Followers About Following

Representation Power of functions

P Parveen Khurana Jan 7, 2020 · 8 min read

This article covers the content discussed in the Representation Power of Functions module of the Deep Learning course and all the images are taken from the same module.

So far we have seen three models: MP Neuron, Perceptron and Sigmoid Neuron but none of them were able to deal with the non-linearly separable data.

In this article, we discuss the Representation power of functions which will help us understand why we need complex functions as our model.

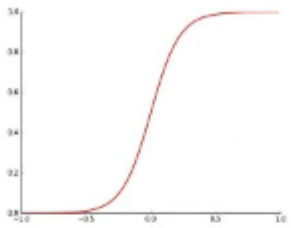
We need to find continuous functions and the reason for that is very simple as we are going to use the Gradient Descent Algorithm where we move in a direction opposite to the gradient while updating the parameters as per the below update rule.

$$b_{t+1} = b_t - \eta \Delta b_t$$

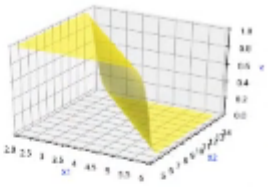
$$\Delta w_t = \frac{\partial L}{\partial w}$$

$$\Delta b_t = \frac{\partial L}{\partial b}$$

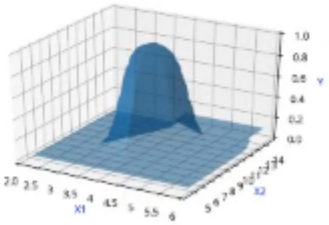
Now to compute the derivative with respect to the parameters, the function must be differentiable so that's why we want to have a continuous function. Example of continuous functions:



$$\hat{y} = \frac{1}{1+e^{-(2*x_1+5)}}$$

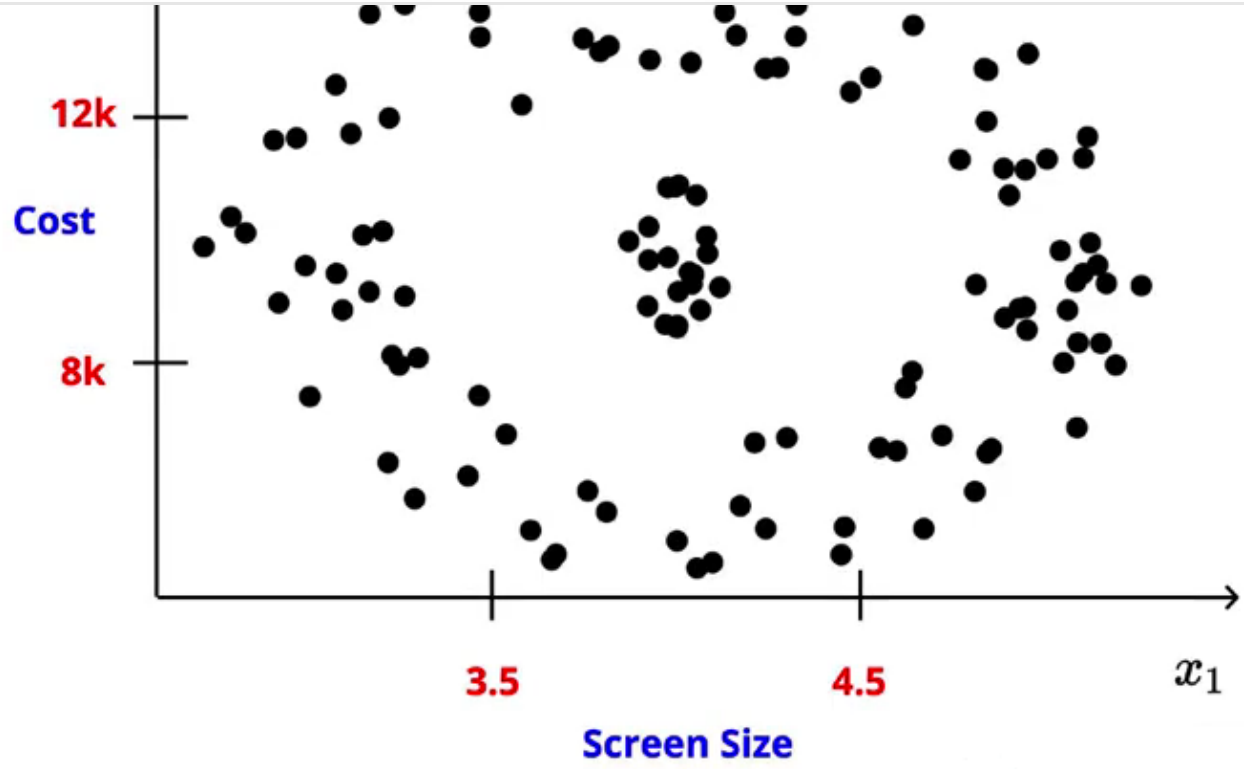


$$\hat{y} = \frac{1}{1+e^{-(-2*x_1+2*x_2+20)}}$$



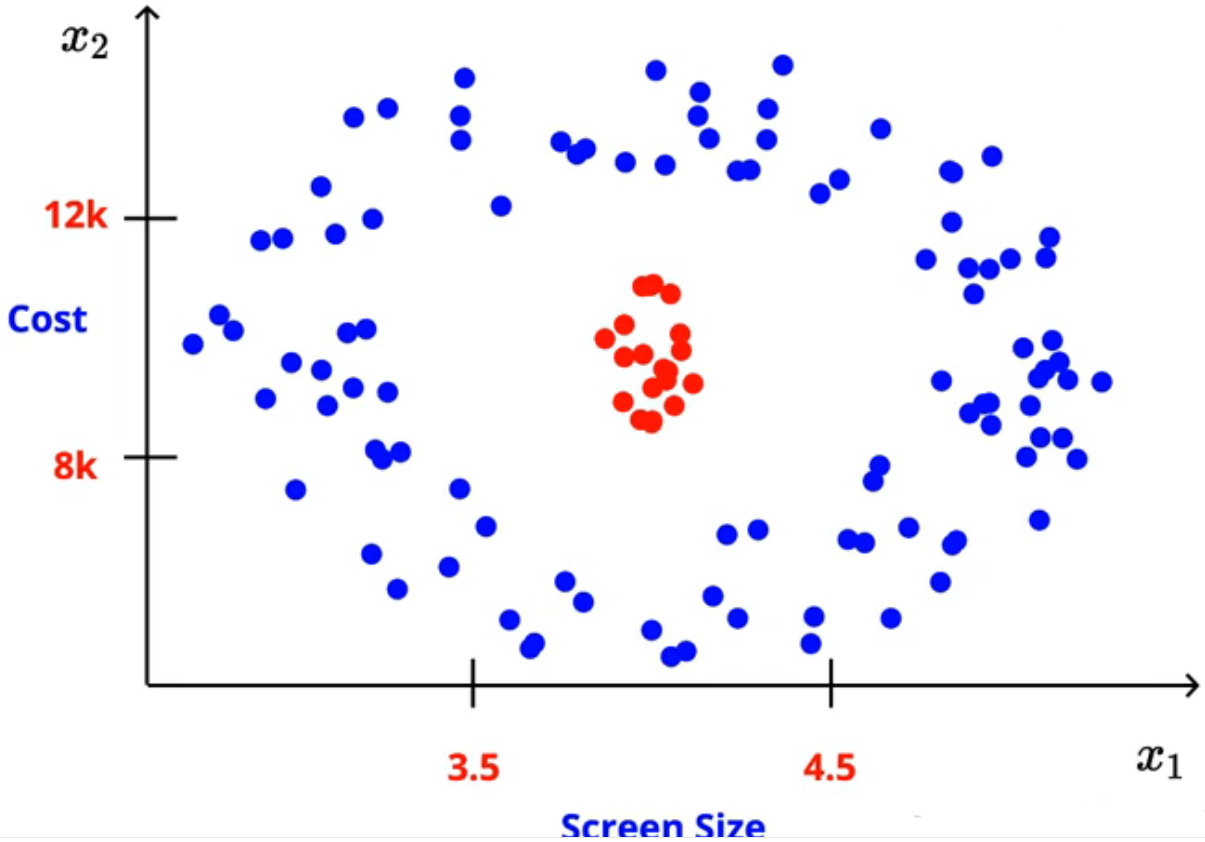
$$\hat{y} = sig_1(sig_2(x_1, x_2), sig_3(x_1, x_2), sig_4(x_1, x_2))$$

We need complex functions for modeling complex relations. Let's take an example where we have two features: Cost Price and screen size of a phone



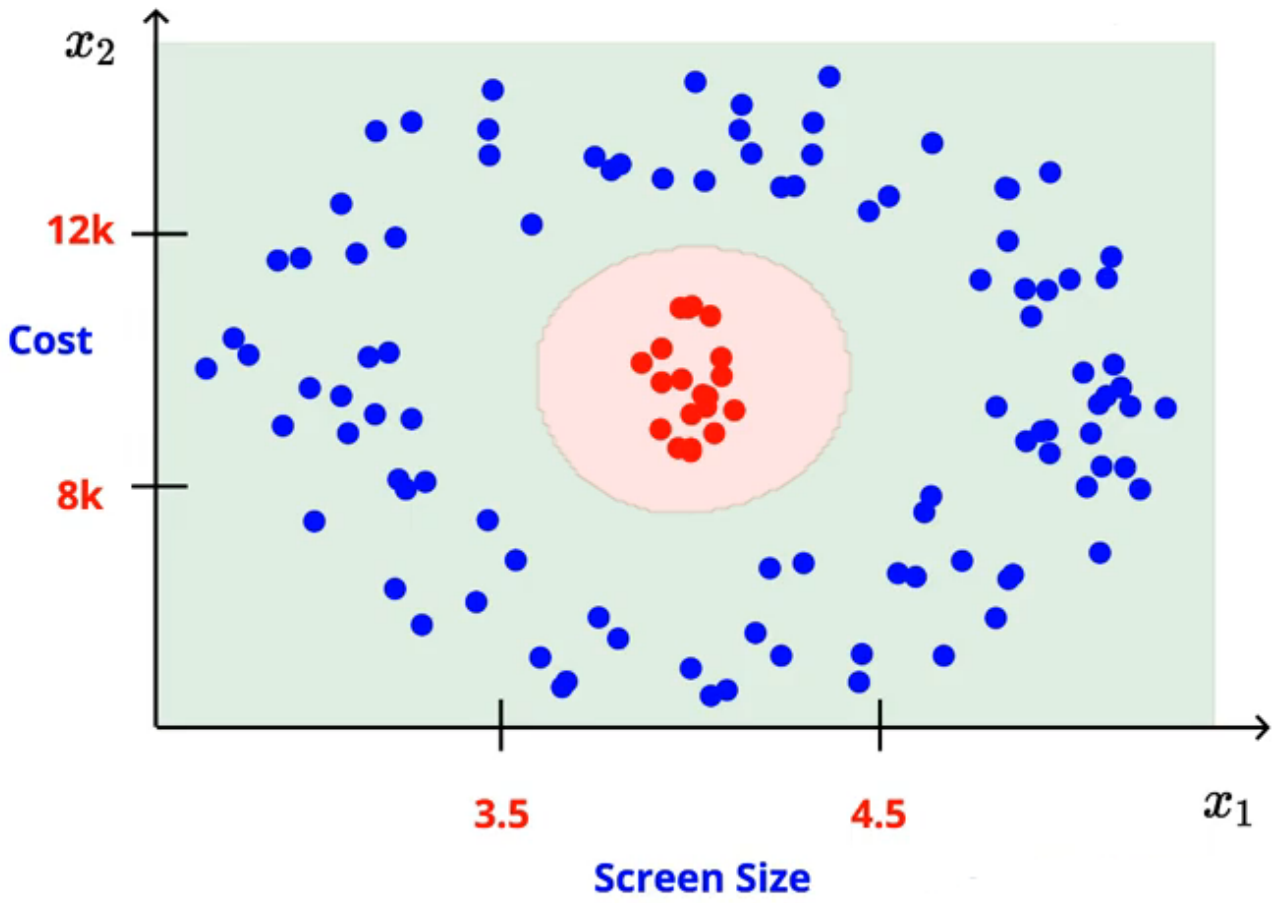
Every point corresponds to one phone.

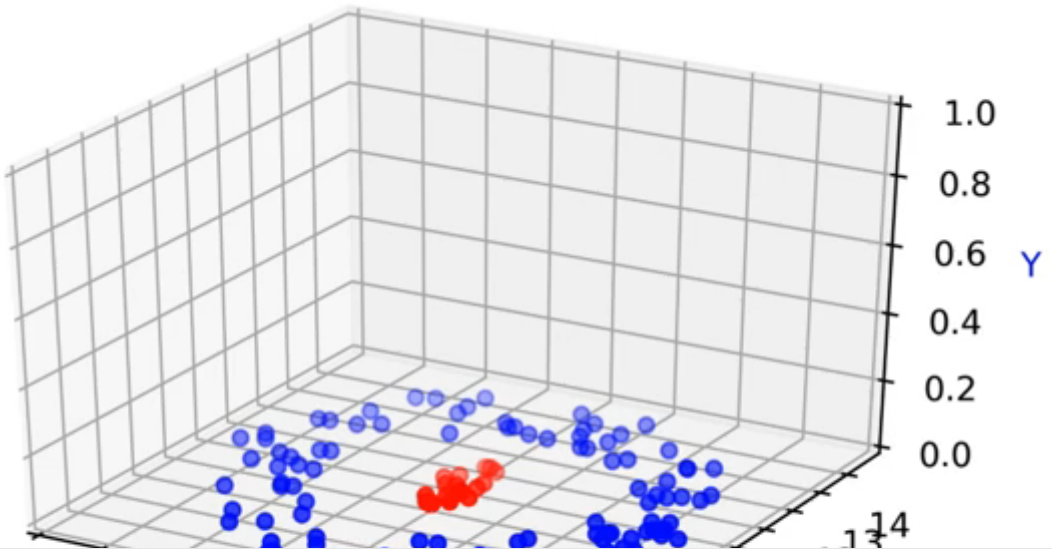
Let's say we like a phone whose screen size is between 3.5 to 4.5 and the price range is from 8k to 12k. So, data would look like:



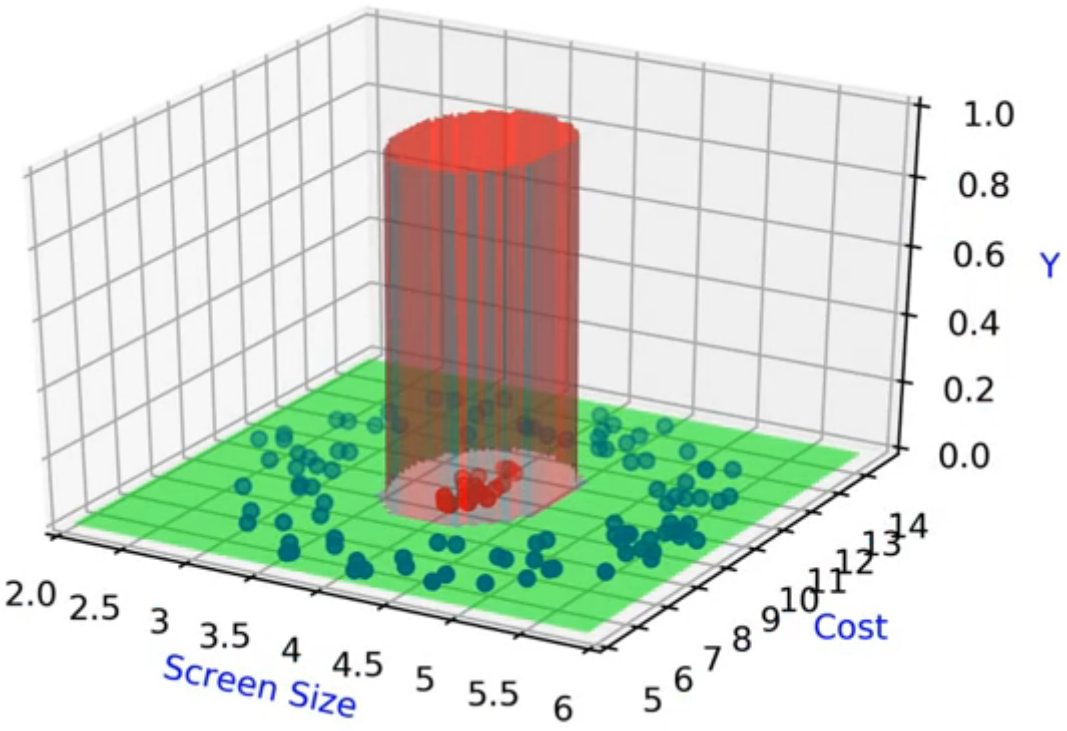
As is clear, this is non-linear data. We cannot draw a line in any way such that it can separate the red points from the blue points.

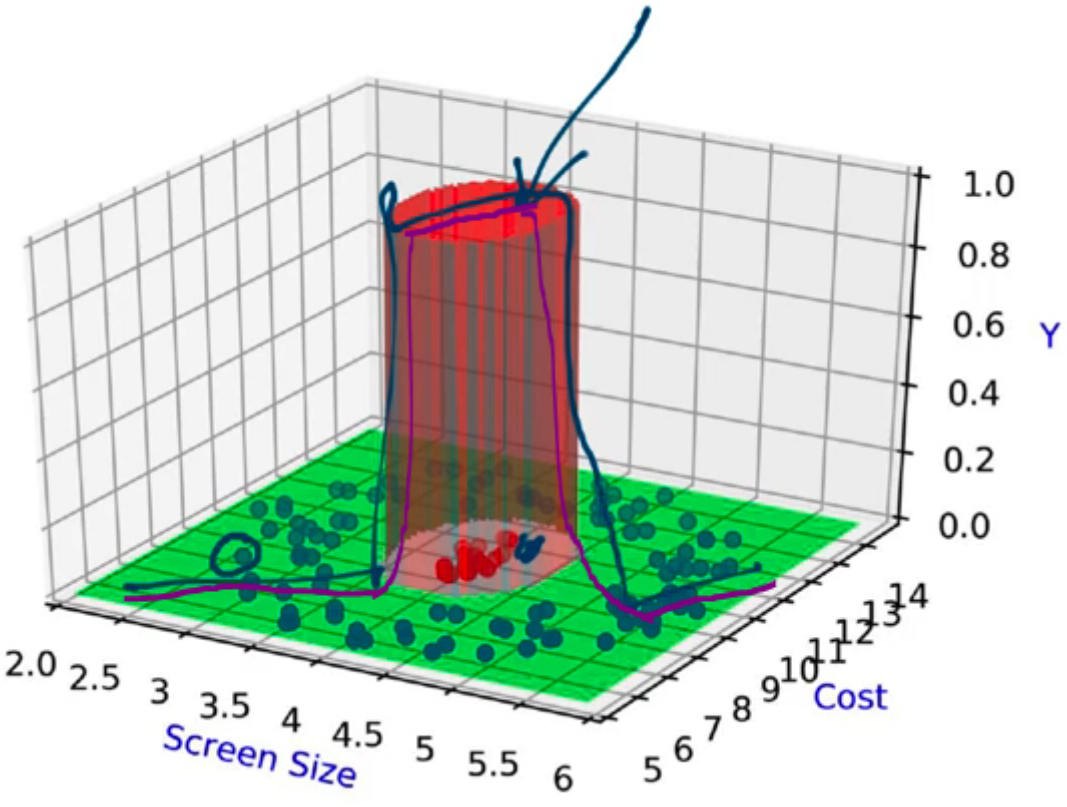
We want the model to be such that it gives an output of 0 for all the points in the green region(all the blue points) and it outputs 1 for all the points in the red region(all the red points) in the below image:



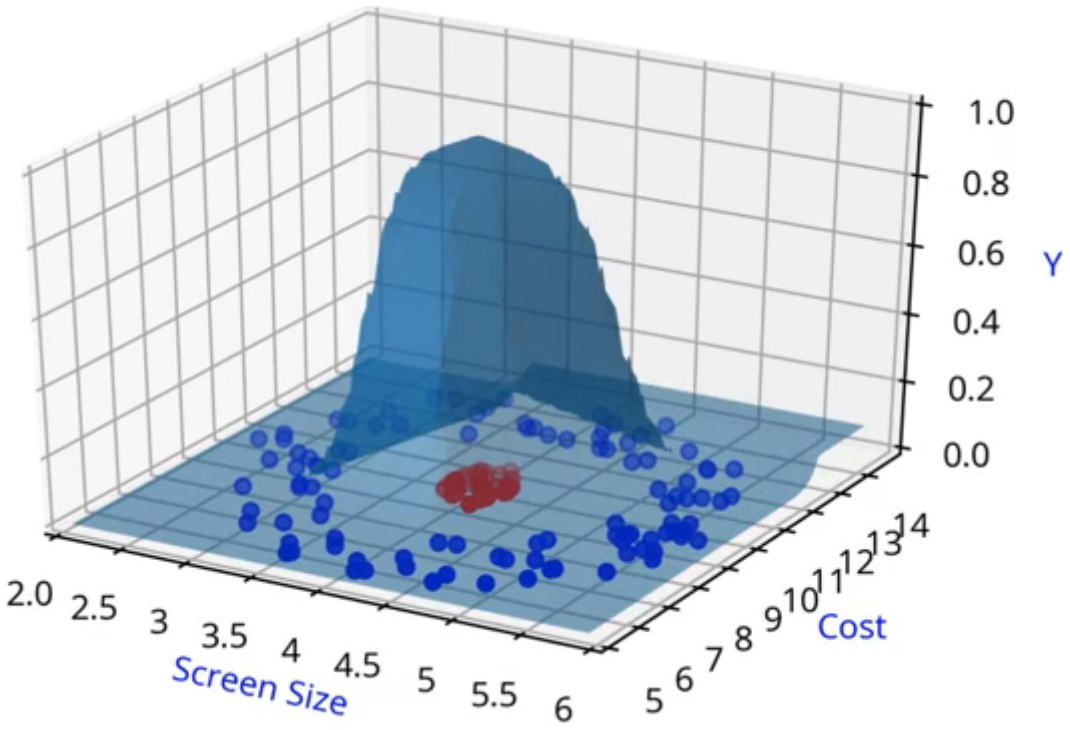


We want the function output to be 1 for the red points and the function output to be 0 for the blue points that mean we want our function to be of the following form:





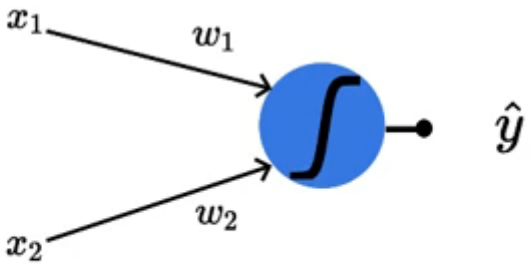
is a very smooth function, a function of the following form:

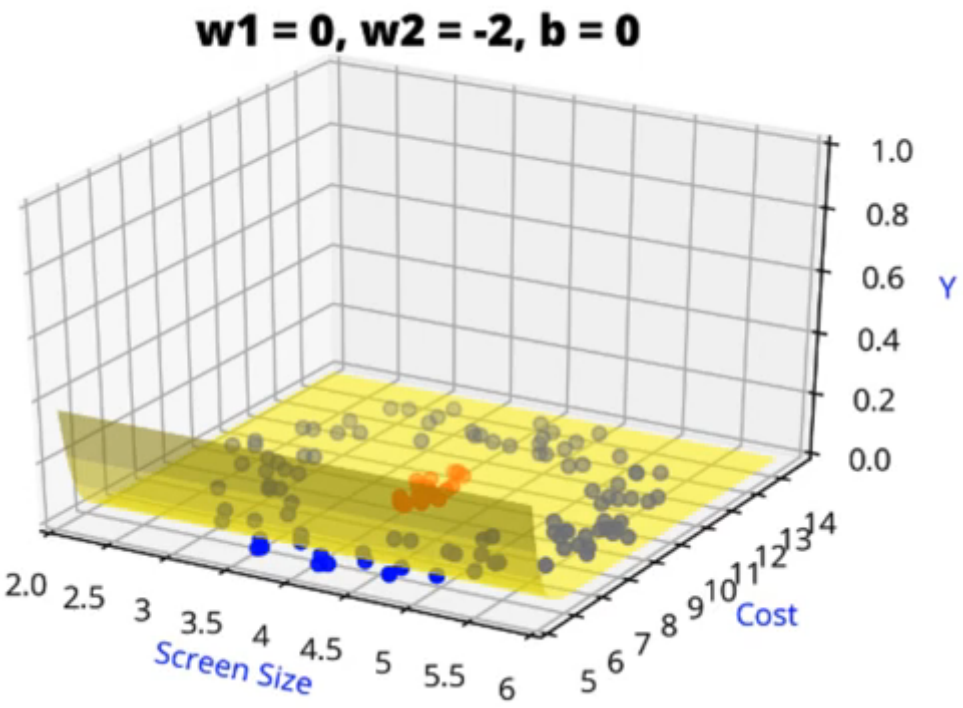


This is one of the reasons why we need complex functions because a lot of real-world data would require functions of above form and the above function is not like a Sigmoid Neuron(S-shaped), it's not like an MP Neuron or Perceptron(Linear), so we need different family of functions, we need more complex functions than what we have seen so far.

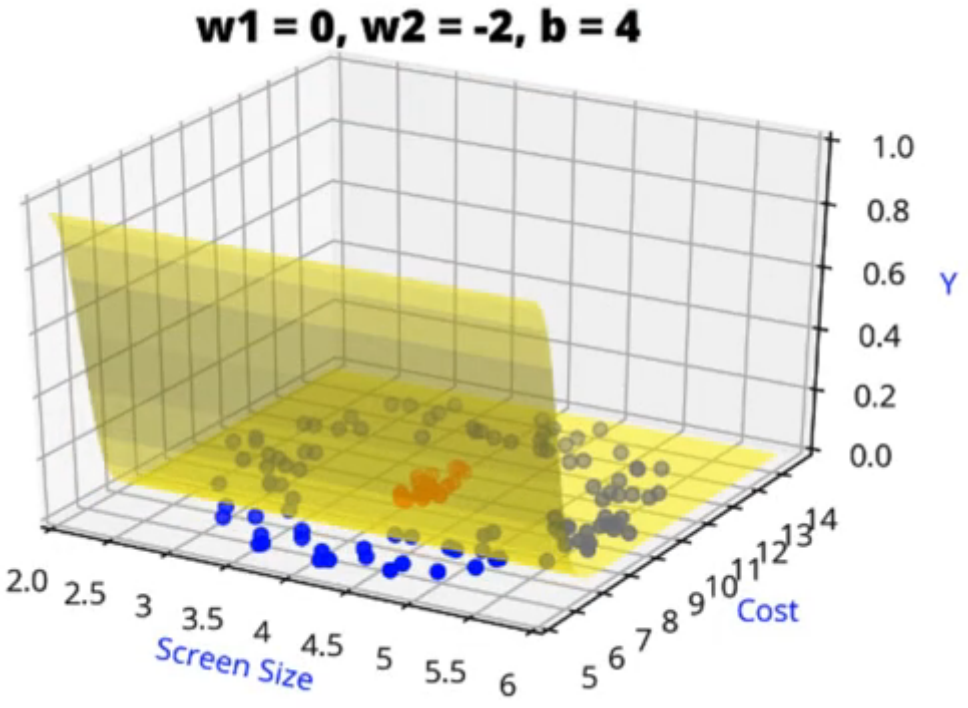
Below is the Sigmoid function that we have:

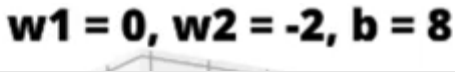
$$\hat{y} = \frac{1}{1+e^{-(w_1*x_1+w_2*x_2+b)}}$$

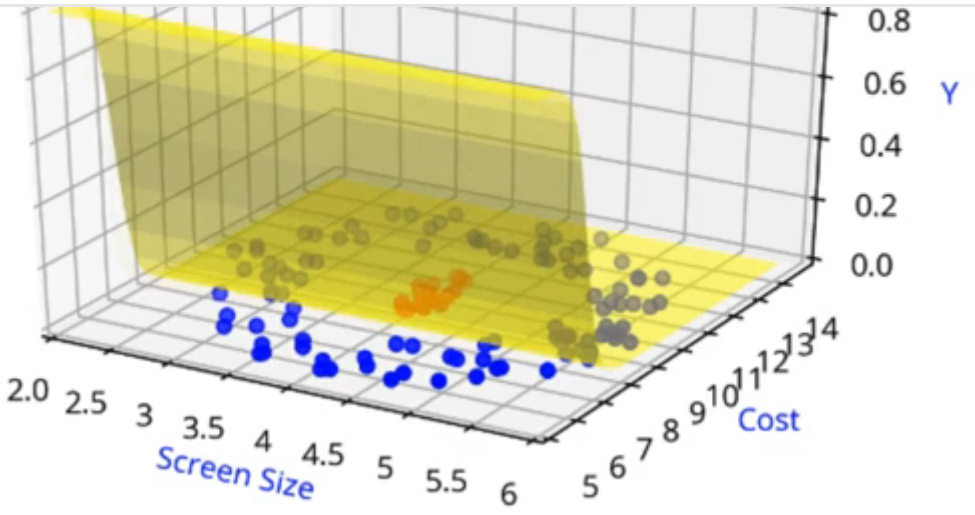


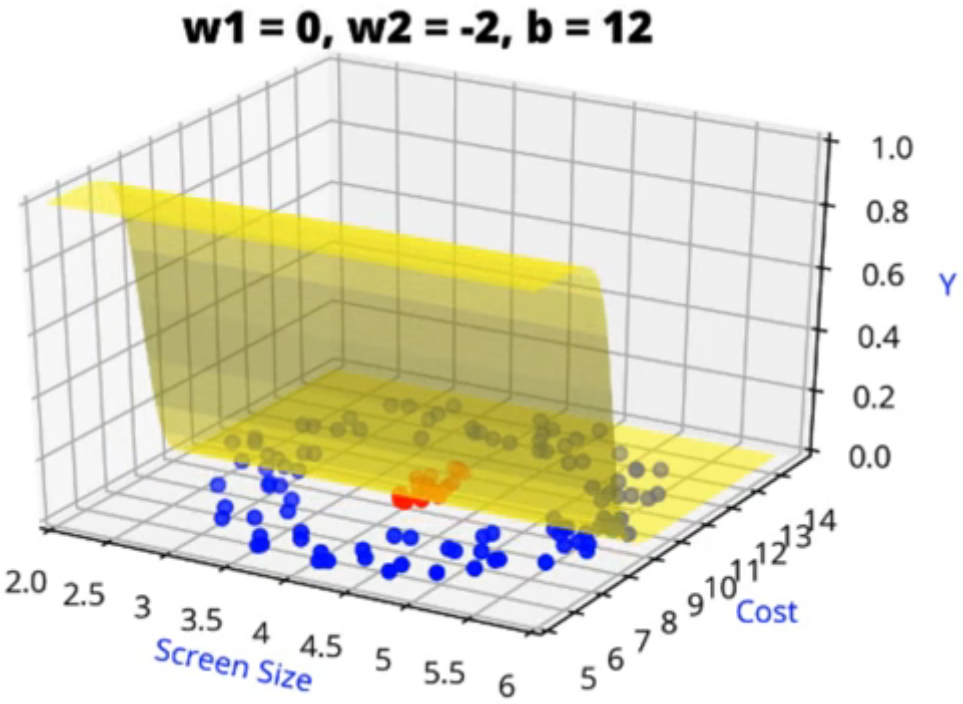


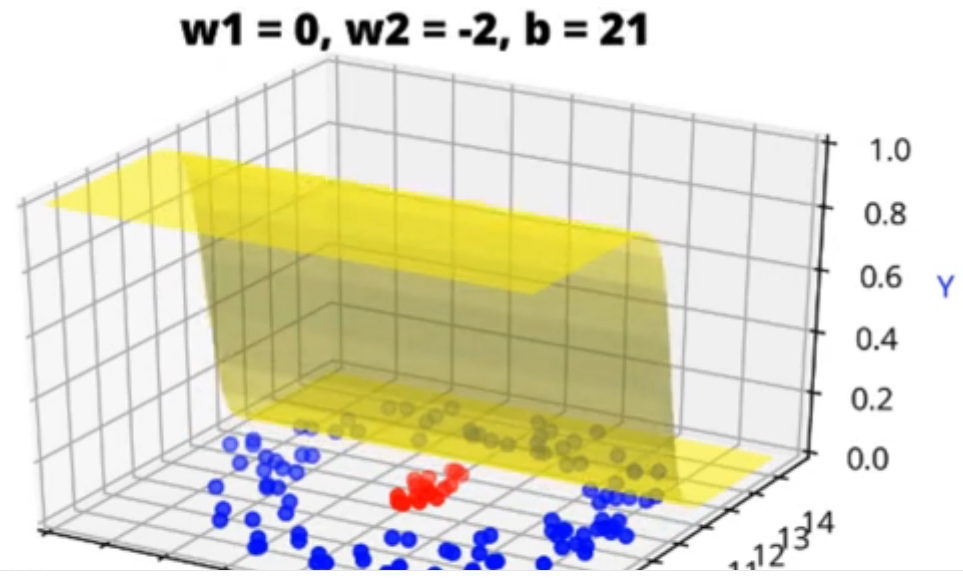
Now no matter what we set the values of parameters, we are not going to get a surface that can exactly fit the data.

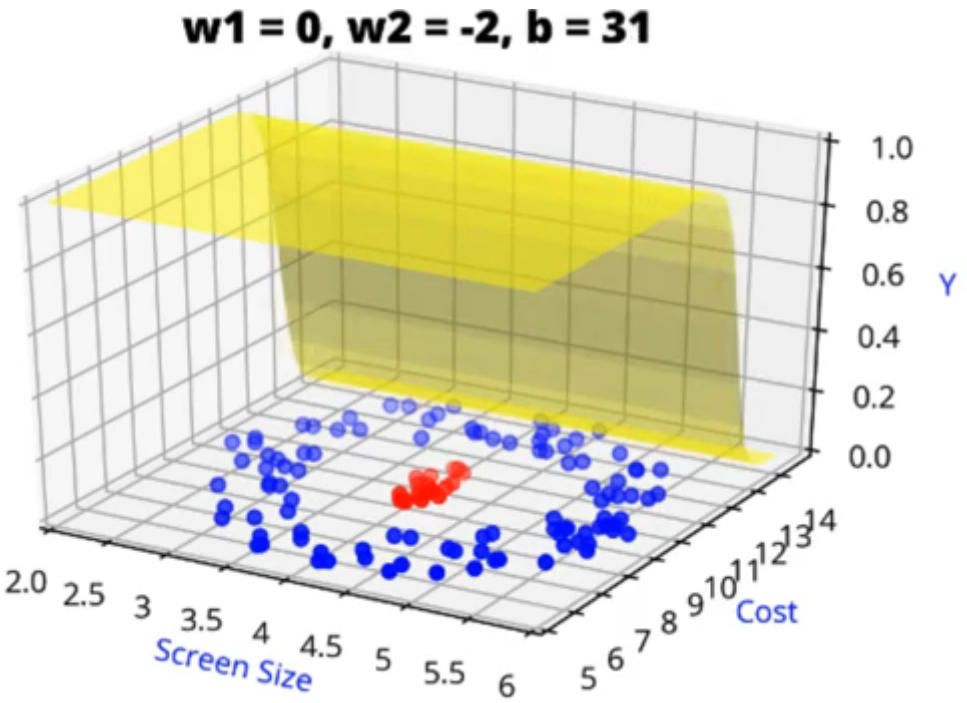


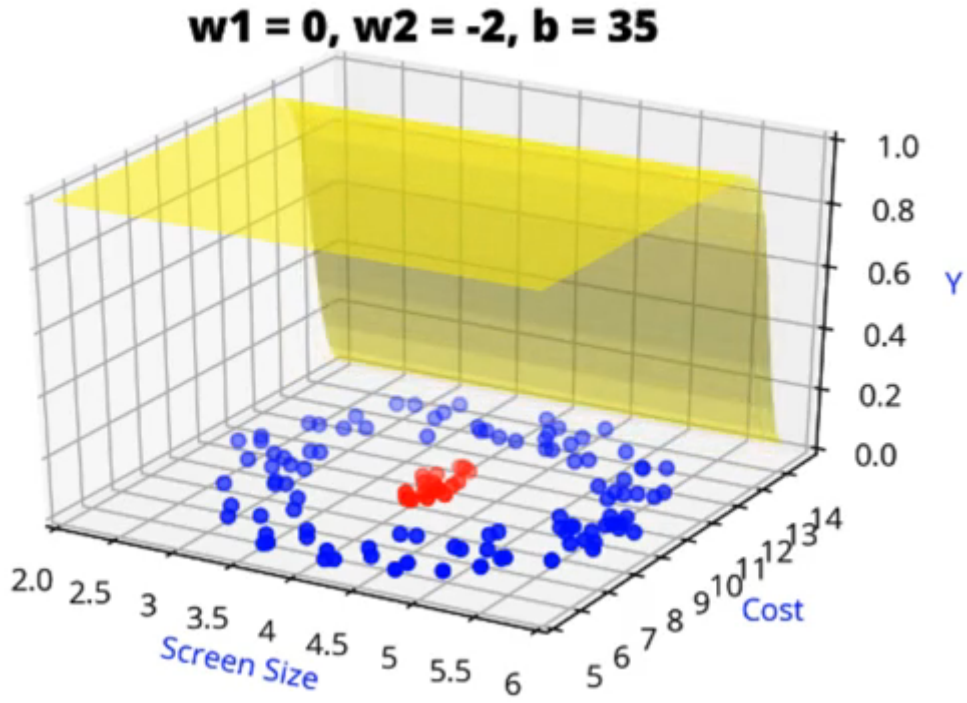


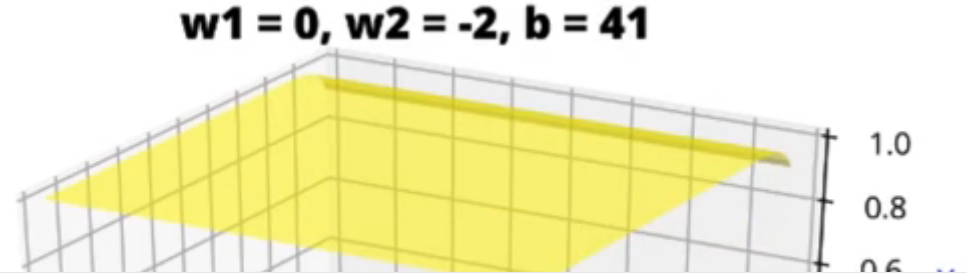

0.8
0.6
Y
0.4
0.2
0.0
2.0 2.5 3 3.5 4 4.5 5 5.5 6
Screen Size
5 6 7 8 9 10 11 12 13 14
Cost

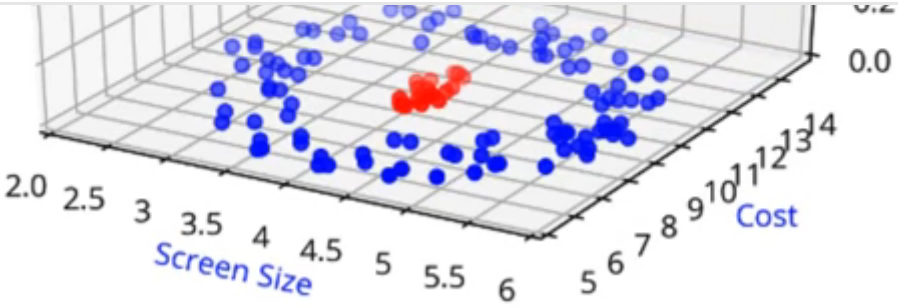

w1 = 0, w2 = -2, b = 12
1.0
0.8
0.6
Y
0.4
0.2
0.0
2.0 2.5 3 3.5 4 4.5 5 5.5 6
Screen Size
5 6 7 8 9 10 11 12 13 14
Cost


w1 = 0, w2 = -2, b = 21
1.0
0.8
0.6
Y
0.4
0.2
0.0


w1 = 0, w2 = -2, b = 31
1.0
0.8
0.6
Y
0.4
0.2
0.0
2.0 2.5 3 3.5 4 4.5 5 5.5 6
Screen Size
5 6 7 8 9 10 11 12 13 14
Cost


w1 = 0, w2 = -2, b = 35
1.0
0.8
0.6
Y
0.4
0.2
0.0
2.0 2.5 3 3.5 4 4.5 5 5.5 6
Screen Size
5 6 7 8 9 10 11 12 13 14
Cost

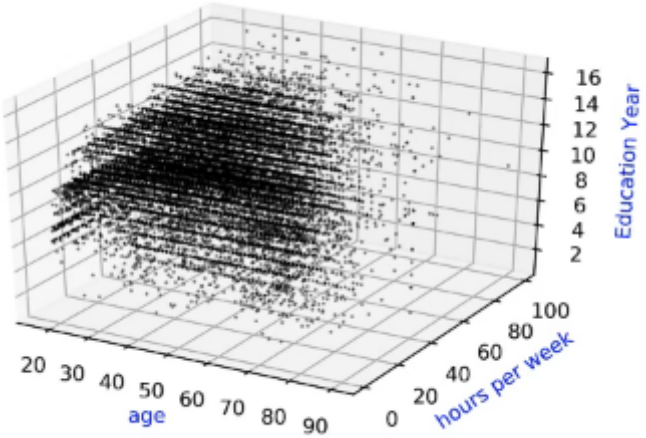

w1 = 0, w2 = -2, b = 41
1.0
0.8



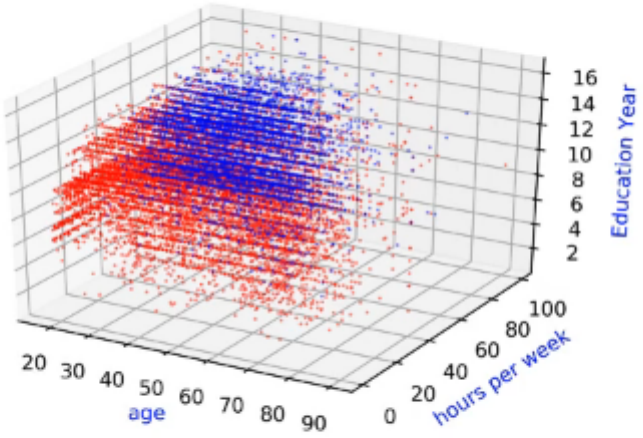
So, sigmoid function, although it looks complex is not something that can serve the purpose here.

The point to consider is that most of the real-world data is not linearly separable, so we are going to need a complex function to fit data. For example: consider the below scenario where the census data is given and we are supposed to make sense of it and predict whether the person's income is greater than 50k or not based on many many features like age, work hours per week, number of years of education and so on.

Age	hour/week	Education year		Income
90	40	9		0
54	40	4	.....	0
74	20	16		1
45	35	16		1
⋮	⋮	⋮	⋮	⋮

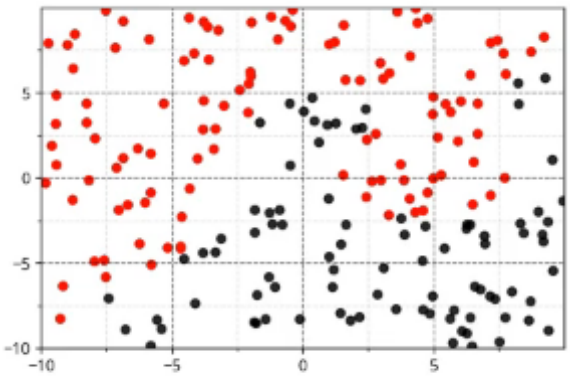


So, if we plot this data based on the true outcome/ output, we would have:

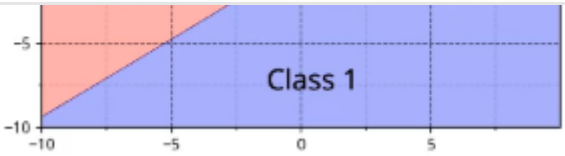


We can easily see that the above data is not linearly separable and even if we use sigmoid, it's not going to serve our purpose here.

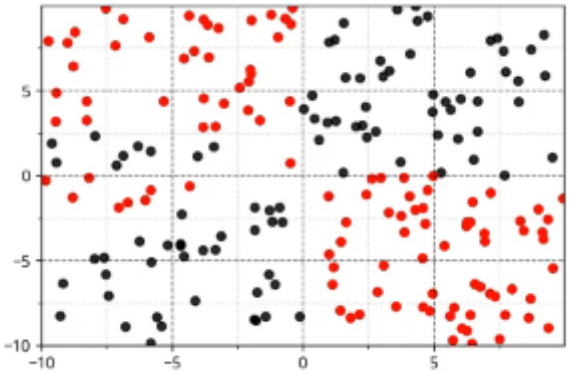
For the below plot



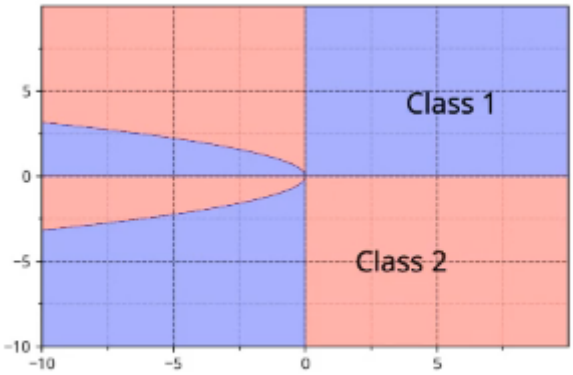
we would like our function to be something like this:



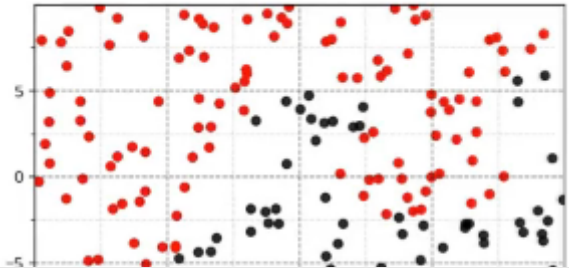
Let's look at another case where we have the data like this:

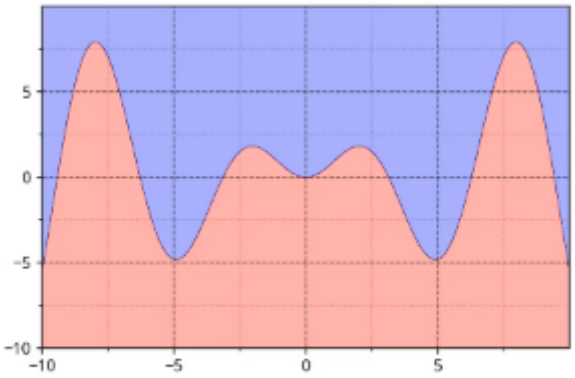


And for this data, we need our function to be something like the below:



Another example:



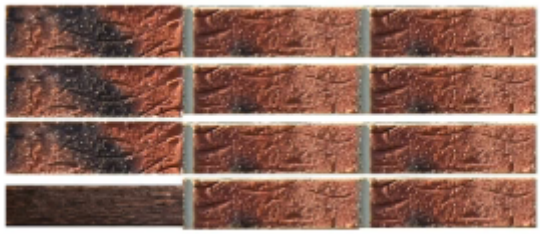


How do we even come up with such Complex functions?

Let's take the analogy of building a house. We could think of it as a very complex output, we are starting from scratch and we want to build something very complex.

We don't start by building the whole house at once, what we do is that we start with the basic building block

output



So, now everything which is participating in it, it's a very simple and basic thing which is participating in the construction of this output, they are the bricks, we just combine them in an effective way so that we get this complex output which is a house.

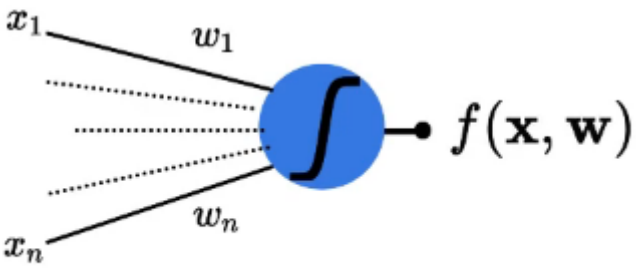
And now we can have different types of houses for example:

Again we can start with the same building block, just arrange it in a different way, add some binding material in between and we will get this output

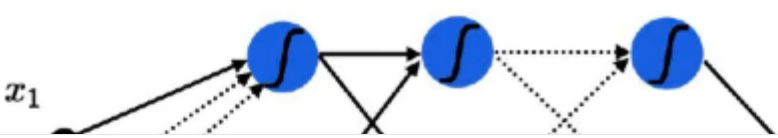
In our case, instead of bricks and houses, what we are interested is in complex functions and the sigmoid neuron acts as the mathematical equivalent to a brick, it is the building block for complex functions.

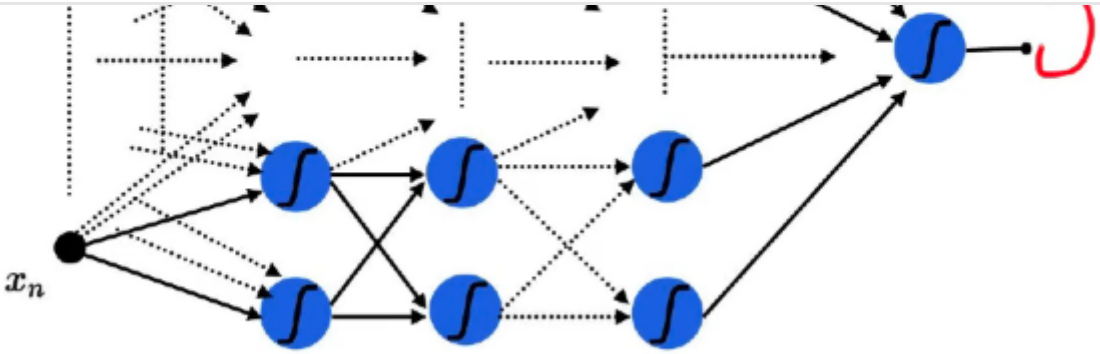
$$f(x_1, .., x_n) = \frac{1}{1+e^{-(w_1 * x_1 + ... + w_n * x_n + b)}}$$

$$f(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-(w*x+b)}}$$



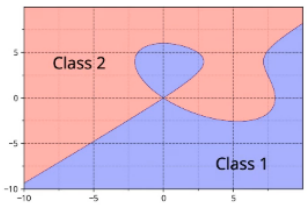
Now we can arrange these bricks in a manner that we get the desired output.

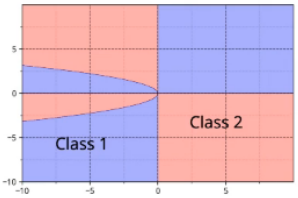


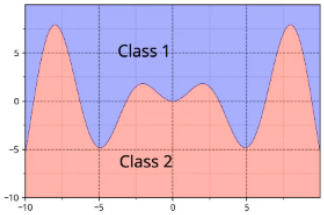


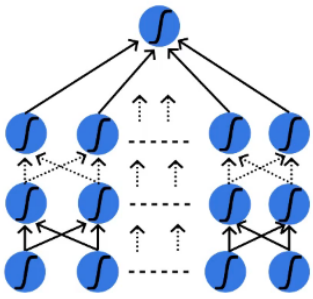
The **final output y_hat** would be a **function of the inputs x1, x2, …., xn** and not only that, **it's going to be a very complex function of these inputs because these inputs are going through multiple transitions, many such transitions at every layer, then the output of those transitions are being combined, many of them are being combined, and all dotted edges/arrows that we have in the above image connecting two neurons, all those have some weights across multiple layers.**

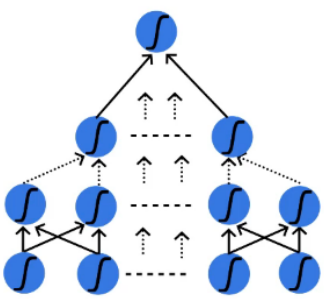
By coming up with different configurations, no matter what the true function is, we would be able to approximate it. **In other words, a Deep Neural Network with a certain number of hidden layers should be able to approximate any function that exists between the input and the output. This is also known as Universal Approximation Theorem and it conveys the representational power of a Deep Neural Network.**

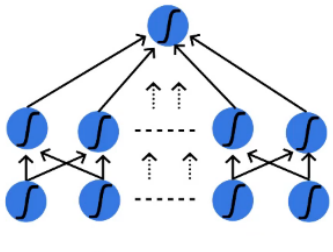












$$f(x_1, .., x_n) = \frac{1}{1+e^{-(w_1 * x_1 + ... + w_n * x_n + b)}} \equiv \quad f(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-(w*x+b)}} \equiv$$

of the model), we try out all of the configurations, now whichever configuration gives us the minimum loss, that is the one best approximating the relationship between the true output and the predicted output.

And for most of the problem statements, say for example for object detection, a good number of standard configurations are available in the public domain.

Take-aways:

We are dealing with real inputs and we have a complex task where the relation between the input and the output is non-linear.

We need functions which can capture/learn the non-linear surfaces and are differentiable; we use sigmoid neuron as a basic building block for such functions and we combine a number of such neurons to build the network which then can approximate the relation between input and output. And this network of functions would be differentiable as the basic block is differentiable.

Machine Learning Deep Learning Artificial Intelligence Artificial Neural Network

Deep Neural Networks